QBUS6850
Lecture 3
Model & Feature Selection

BUSINESS SCHOOL

QBUS6850 Team

THE UNIVERSITY OF SYDNEY

Topics covered

- Logistics regression
- Intuition of regularization
- Regularized Linear Regressions: Ridge, LASSO, Elastic Net
- Feature Extraction

References

- Bishop (2006), Chapters 3.1.4; 4.3.2
- Hastie et al. (2001), Chapter 3.4, Chapter 7.7-7.10
- James et al., (2014), Chapters 4.3; 6.2

THE UNIVERSITY OF SYDNEY

Learning Objectives

- ❑ Understand the intuition of regularization
- ❑ Review how Ridge regression, LASSO regression and Elastic net work
- ❑ Understand the differences between various regularized regressions
- ❑ Understand difference between regression and classification
- ❑ Be able to calculate the predicting probability with logistic regression
- ❑ Understand different types of features and feature extraction
- ❑ Be able to extract features for text data
- ❑ Understand Cross Validation and Be able to conduct Cross Validation

Classification

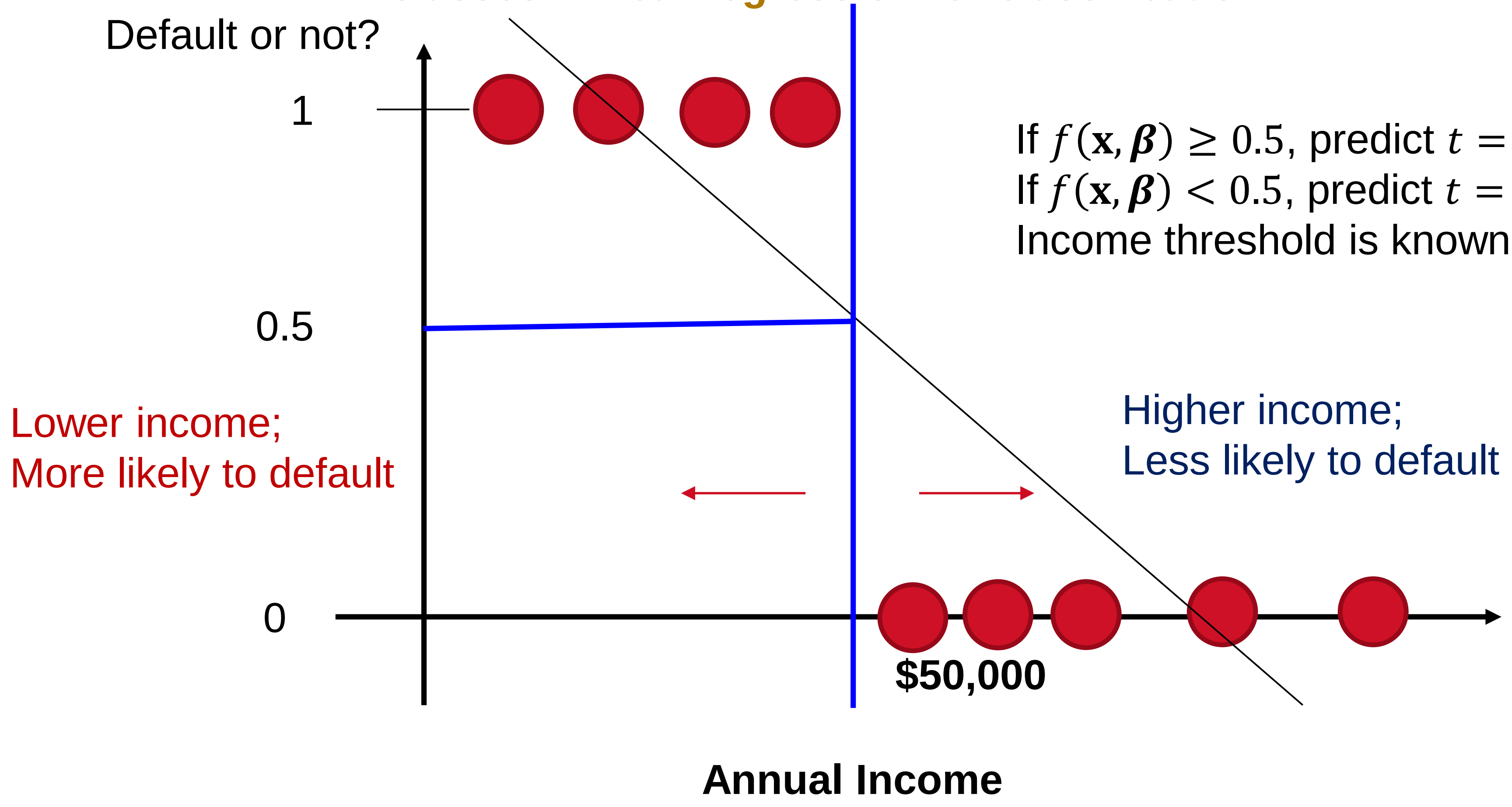
- ❑ The linear regression was used as an example to show the machine learning workflow
- ❑ The major ingredients are data, a model, and a criterion (objective)

  What are they in linear regression?
- ❑ In regression, the target t was continuous numeric value; however in many applications, we wish to predict class instead of an amount
- ❑ When the target is categorical, we call the regression as a classification
- ❑ For classification, how shall we choose a model? how shall we design a criterion to measure the “error” between the observation and the model prediction?
- ❑ We will look at the logistic regression as an example

Classification

Supervised learning with categorical response: classification

2 classes. Linear regression for classification?

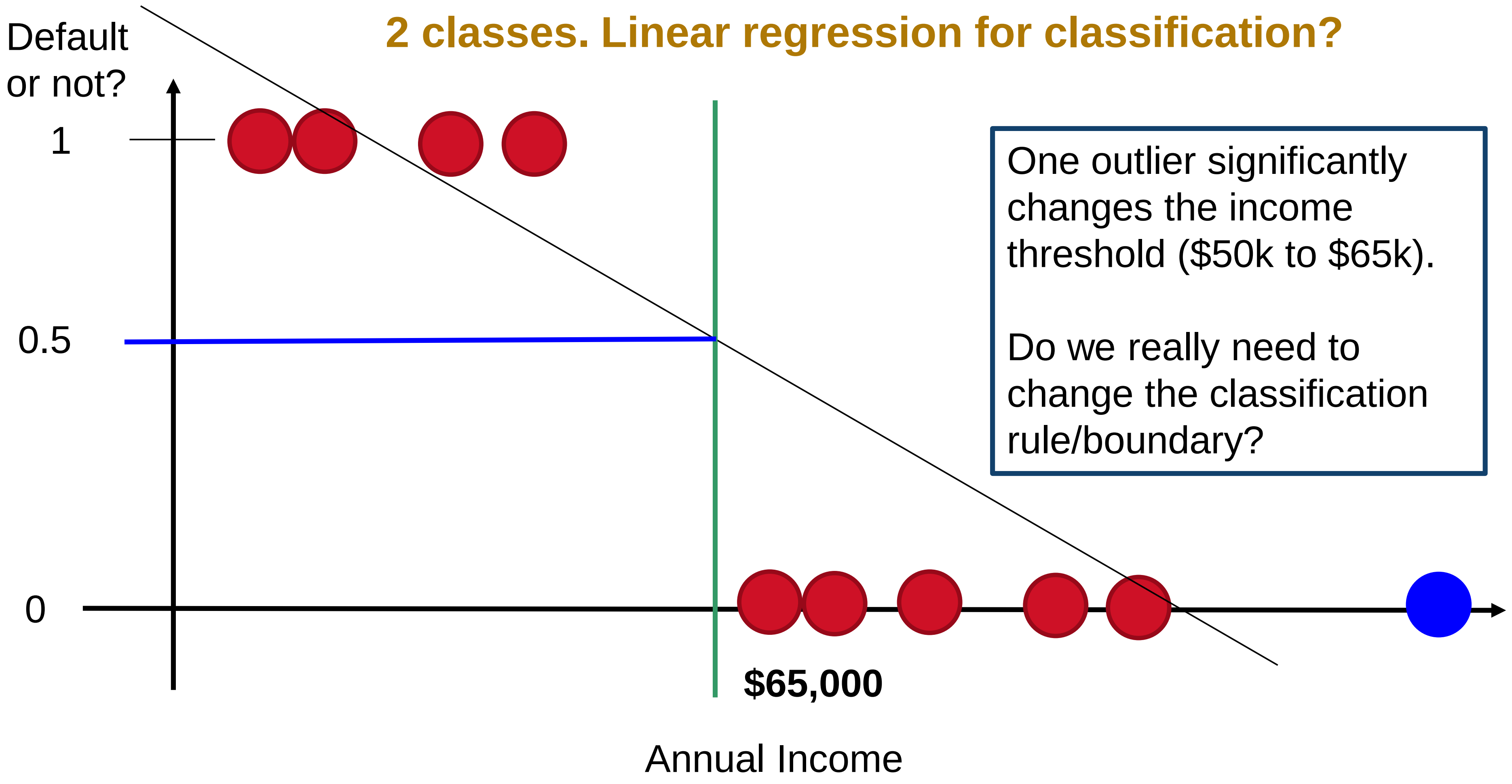


If $f(\mathbf{x}, \boldsymbol{\beta}) \geq 0.5$, predict $t = 1$;
If $f(\mathbf{x}, \boldsymbol{\beta}) < 0.5$, predict $t = 0$.
Income threshold is known.

Classification

Supervised learning with categorical response

2 classes. Linear regression for classification?



One outlier significantly changes the income threshold ($50k to $65k).

Do we really need to change the classification rule/boundary?

Logistic Regression

Re-coding Target

- In fact, there is no numeric target value in classification. We manually encode it as 1 (class A) or 0 (class B).
- Can we re-code "Class A" as (1, 0) and "Class B" as (0, 1)? That is, for each case, we have two target values or the target is a vector $\mathbf{t} = (t_1, t_2)$ [Please recall our notation in Lecture 2 where $m = 2$]
- Hence we shall have two models $f_A(\mathbf{x}, \boldsymbol{\beta}) \rightarrow t_1$ and $f_B(\mathbf{x}, \boldsymbol{\beta}) \rightarrow t_{\mathbf{2}}$
- How shall we measure the error between them? [We need a learning criterion or objective]
- It seems, for our coding (1, 0) and (0,1) for classes A and B, we have $t_1 + t_2 = 1$, and $0 \leq t_1 \leq 1, 0 \leq t_2 \leq 1$.
- Can we say $\mathbf{t} = (t_1, t_2)$ is a Bernoulli distribution with a parameter t_1?
- Hence, each training target (class A or class B) becomes an "extreme" Bernoulli distribution either (1, 0) or (0, 1)

Developing New Objective

- Hence, each training target (class A or class B) becomes an "extreme" Bernoulli distribution either (1, 0) or (0, 1)
- Two models $f_A(\mathbf{x}, \boldsymbol{\beta}) \rightarrow t_1$ and $f_B(\mathbf{x}, \boldsymbol{\beta}) \rightarrow t_2$ shall aim to predict the Bernoulli parameter, or $f_A(\mathbf{x}, \boldsymbol{\beta})$ should be the probability for case $\mathbf{x}$ to be class A and $f_B(\mathbf{x}, \boldsymbol{\beta})$ should be the probability for case $\mathbf{x}$ to be class B.
- Three conditions: $0 \leq f_A(\mathbf{x}, \boldsymbol{\beta}) \leq 1$ and $0 \leq f_B(\mathbf{x}, \boldsymbol{\beta}) \leq 1$, and $f_A(\mathbf{x}, \boldsymbol{\beta}) + f_B(\mathbf{x}, \boldsymbol{\beta}) = \mathbf{1}$
- That is $(f_A(\mathbf{x}, \boldsymbol{\beta}), f_B(\mathbf{x}, \boldsymbol{\beta}))$ is a Bernoulli distribution for each case $\mathbf{x}$ too.
- Suppose we already have models satisfying the above conditions, now the question becomes how we tell the Bernoulli $(f_A(\mathbf{x}, \boldsymbol{\beta}), f_B(\mathbf{x}, \boldsymbol{\beta}))$ is close to Bernoulli (1, 0) [if $\mathbf{x}$ is class A] or Bernoulli (0, 1) [if $\mathbf{x}$ is class B]

Cross Entropy Objective

- Our simple example has demonstrated that simply measuring the squared errors is not a good way

$$(f_A(\mathbf{x}, \boldsymbol{\beta}) - t_1)^2 + (f_B(\mathbf{x}, \boldsymbol{\beta}) - t_2)^2$$

- Although a Bernoulli distribution is represented by a 2D vector, they are special: two components are between 0 and 1, and the sum of them is 1.
- To measure the "distance" between distributions, we use either the so-called Kullback-Leibler divergence, or the so-called cross entropy. For two Bernoulli distributions $(f_A(\mathbf{x}, \boldsymbol{\beta}), f_B(\mathbf{x}, \boldsymbol{\beta}))$ and $\mathbf{t} = (t_1, t_2)$, the cross entropy is defined as

$$-t_1 \log(f_A(\mathbf{x}, \boldsymbol{\beta})) - t_2 \log(f_B(\mathbf{x}, \boldsymbol{\beta})))$$

- For all the data we have

$$L(\boldsymbol{\beta}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\left(t_{n1} \log\left(f_A(\mathbf{x}_n, \boldsymbol{\beta})\right) + t_{n2} \log\left(f_B(\mathbf{x}_n, \boldsymbol{\beta})\right)\right)\right]$$

Clarification

- Don't confuse with a number of things here
- Two Classes A and B:
  - can be labelled as 1 and 0 respectively, so we use target value t = 1 or 0
  - can be encoded as (1, 0) and (0, 1) respectively, regarded as hot-one code or Bernoulli distribution. We can focus on the first component 1 and 0, respectively [This is not label, but the probability]
- In both cases, we can simply use one target variable (not a vector) t which takes value of 1 (for class A) and 0 (for class B).
- Similarly we only need focus on $f_A(\mathbf{x}, \boldsymbol{\beta})$ because $f_B(\mathbf{x}, \boldsymbol{\beta}) = 1 - f_A(\mathbf{x}, \boldsymbol{\beta})$. Simply we use $f(\mathbf{x}, \boldsymbol{\beta})$ for $f_A(\mathbf{x}, \boldsymbol{\beta})$, i.e., we need only one model
- Finally the loss is defined as

$$L(\boldsymbol{\beta}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\left(t_n \log\left(f(\mathbf{x}_n, \boldsymbol{\beta})\right) + (1 - t_n)\log\left(1 - f(\mathbf{x}_n, \boldsymbol{\beta})\right)\right)\right]$$

Logistic Function

How can we make a model satisfying $0 \leq f(\mathbf{x}, \boldsymbol{\beta}) \leq 1$?

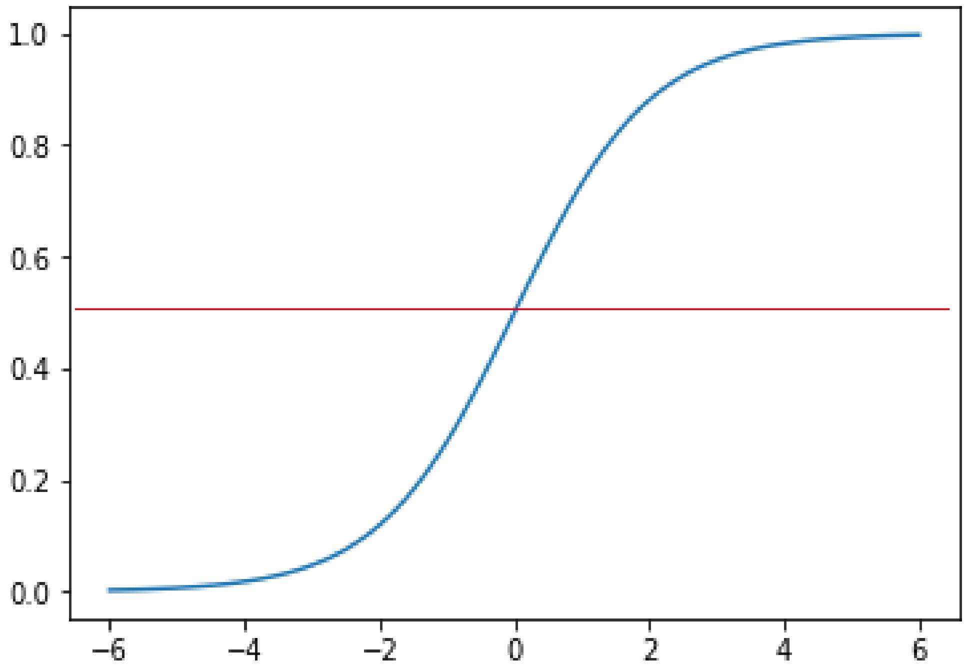
$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$

$$z \in (-\infty, +\infty)$$

$$\sigma(z) \in (0,1)$$

```
x_list= np.linspace(-6,6,100)
y_list= 1/(1+np.exp(-x_list))
plt.plot(x_list, y_list)
```

Logistic Function



Also called **Sigmoid Function**

Logistic Regression

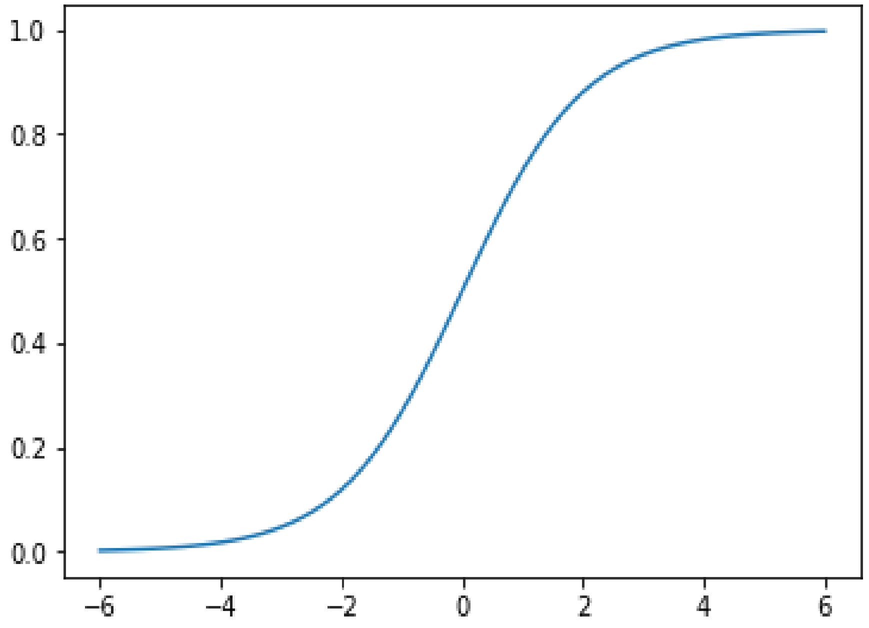
Regression + Logistic Function

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$

$$f(\mathbf{x}, \boldsymbol{\beta}) = \sigma(\mathbf{x}^T\boldsymbol{\beta}) = \frac{1}{1+e^{-\mathbf{x}^T\boldsymbol{\beta}}}$$

If $\mathbf{x}^T\boldsymbol{\beta} \geq 0; f(\mathbf{x}, \boldsymbol{\beta}) \geq 0.5$
predict as class A;
If $\mathbf{x}^T\boldsymbol{\beta} < 0; f(\mathbf{x}, \boldsymbol{\beta}) < 0.5$
predict as class B;

Question:
Why this function is better?
Think about “outlier” few slides before?



Output Interpretation

$f(\mathbf{x}, \boldsymbol{\beta})$ tells us the estimated probability of given input $\mathbf{x}$ being class A, parameterized by $\boldsymbol{\beta}$.

$$P(t = A|\mathbf{x}, \boldsymbol{\beta}) \coloneqq f(\mathbf{x}, \boldsymbol{\beta})$$

$$P(t = B|\mathbf{x}, \boldsymbol{\beta}) \coloneqq 1 - P(t = A|\mathbf{x}, \boldsymbol{\beta}) = 1 - f(\mathbf{x}, \boldsymbol{\beta})$$

Supposed one customer $\mathbf{x}_i$ has annual income of \$120,000

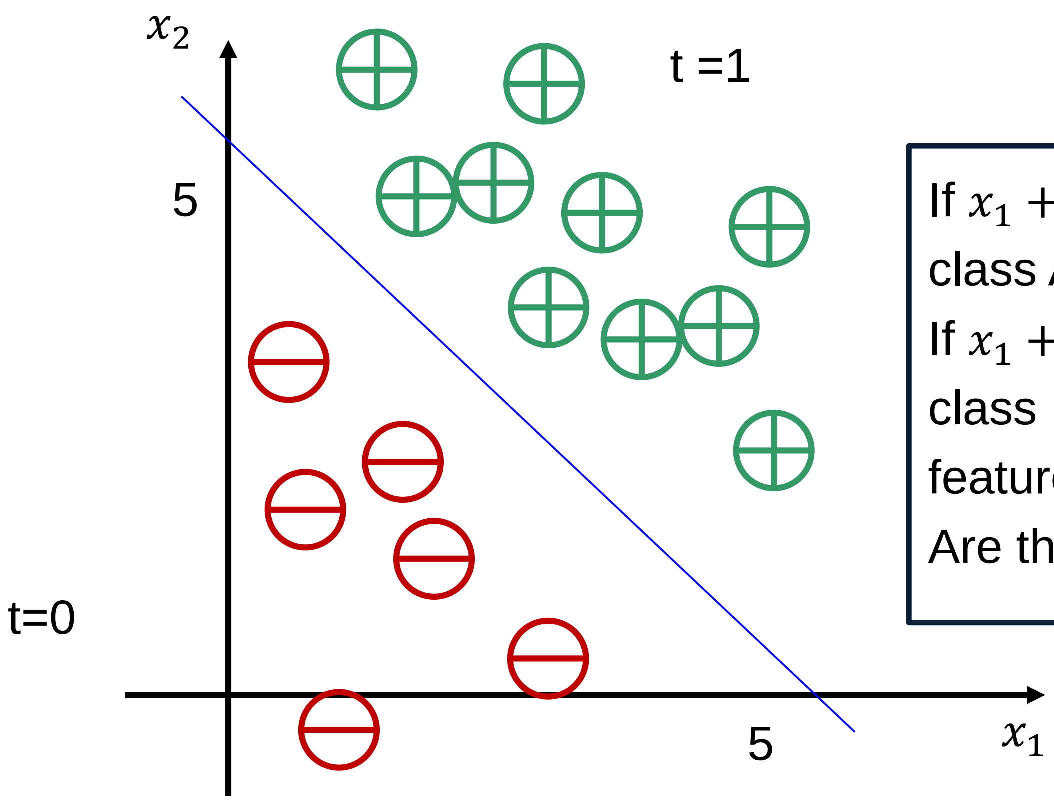
$$f(\mathbf{x}_i, \boldsymbol{\beta}) = 0.1 = 10\%$$

The risk management team of the bank can tell that this customer have 10% probability to default (Class A).

This information is crucial for the bank decision making!

Decision Boundary

$$f(\mathbf{x}, \boldsymbol{\beta}) = \sigma(\mathbf{x}^T \boldsymbol{\beta}) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2) = \sigma(-5 + x_1 + x_2) = 0.5$$

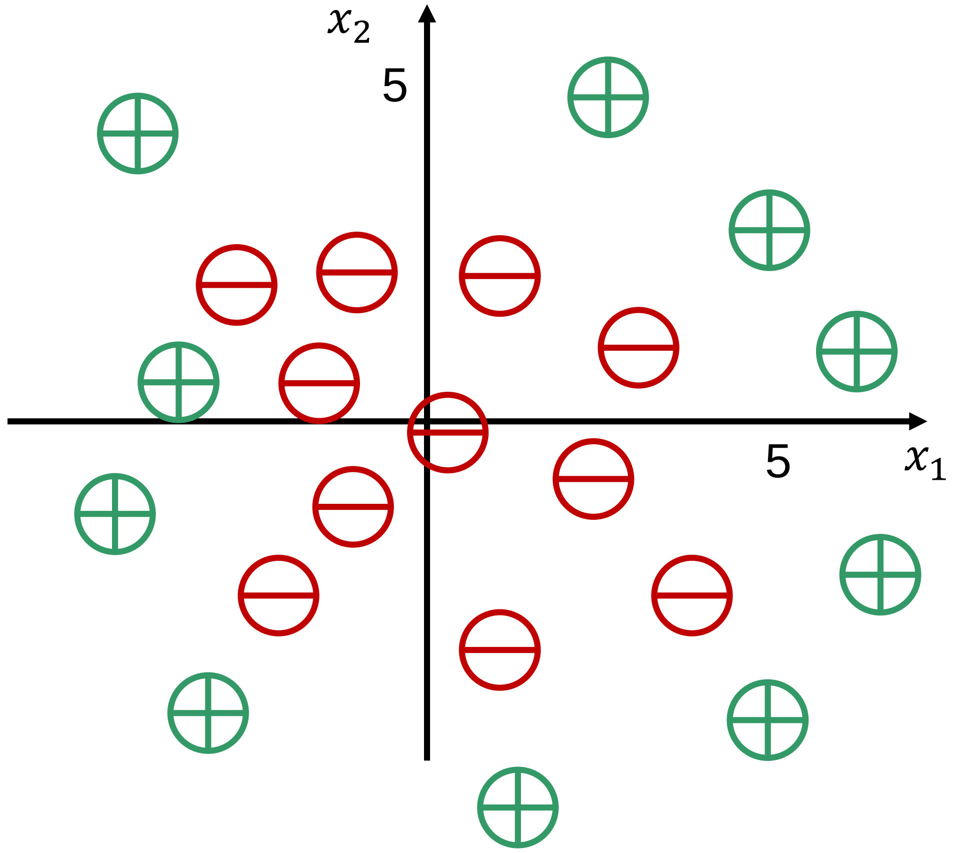


1st and 2nd feature vectors

If $x_1 + x_2 - 5 \geq 0$; predict as class A ($t = 1$);
If $x_1 + x_2 - 5 < 0$; predict as class B ($t = 0$); How many features do we have?
Are the data labelled or not?

Non-linear Decision Boundary

$$f(\mathbf{x}, \boldsymbol{\beta}) = \sigma(\mathbf{x}^T\boldsymbol{\beta}) = \sigma(\beta_0 + \beta_1 x_1^2 + \beta_2 x_2^2) = \sigma(-5 + x_1^2 + x_2^2) = 0.5$$



If $x_1^2 + x_2^2 - 5 \geq 0$; predict $t = 1$;

If $x_1^2 + x_2^2 - 5 < 0$; predict $t = 0$;

How does the decision boundary look like?

Loss Function (Formal)

$$\mathcal{D} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), (\mathbf{x}_3, t_3), \ldots, (\mathbf{x}_N, t_N)\}$$

Once again, collect all the inputs into a matrix **X**, whose size is $N \times (d+1)$ and define the parameter vector

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{bmatrix}$$

N: number of training examples
d: number of features
$\mathbf{x}$**:** “input” variable; **features**
t**:** “output” variable; “target” variable, $\in \{0,1\}$

Note: this is not between 0 and 1

Logistic Regression Loss Function

$$\boxed{L(\boldsymbol{\beta}) = \frac{1}{N}\sum_{n=1}^{N} \boxed{\text{Loss}(f(\mathbf{x}_n, \boldsymbol{\beta}), t_n)}}$$

$$\text{Loss}(\boxed{f(\mathbf{x}_n, \boldsymbol{\beta})}, t_n) = \begin{cases} -\log(f(\mathbf{x}_n, \boldsymbol{\beta})), & t = 1 \\ -\log(1 - f(\mathbf{x}_n, \boldsymbol{\beta})), & t = 0 \end{cases}$$

$$f(\mathbf{x}, \boldsymbol{\beta}) = \sigma(\mathbf{x}^T \boldsymbol{\beta}) = \frac{1}{1 + e^{-\mathbf{x}^T \boldsymbol{\beta}}}$$

Loss Function: Compact Representation

$$L(\boldsymbol{\beta}) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{Loss}(f(\mathbf{x}_n, \boldsymbol{\beta}), t_n)$$

$$\mathrm{Loss}(f(\mathbf{x}_n, \boldsymbol{\beta}), t_n) = \begin{cases} -\log(f(\mathbf{x}_n, \boldsymbol{\beta})), & t = 1 \\ -\log(1 - f(\mathbf{x}_n, \boldsymbol{\beta})), & t = 0 \end{cases}$$

$$\Downarrow$$

$$L(\boldsymbol{\beta}) = -\frac{1}{N}\left[\sum_{n=1}^{N}\left(t_n \log(f(\mathbf{x}_n, \boldsymbol{\beta})) + (1 - t_n)\log(1 - f(\mathbf{x}_n, \boldsymbol{\beta}))\right)\right]$$

This loss function can be derived from statistics using the a methodology called **Maximum Likelihood Estimation (MLE)**

Logistic Regression Summary

- Logistic regression is a special case of **Generalized Linear Models** (GLM)

- logit or sigmoid function is a link function.

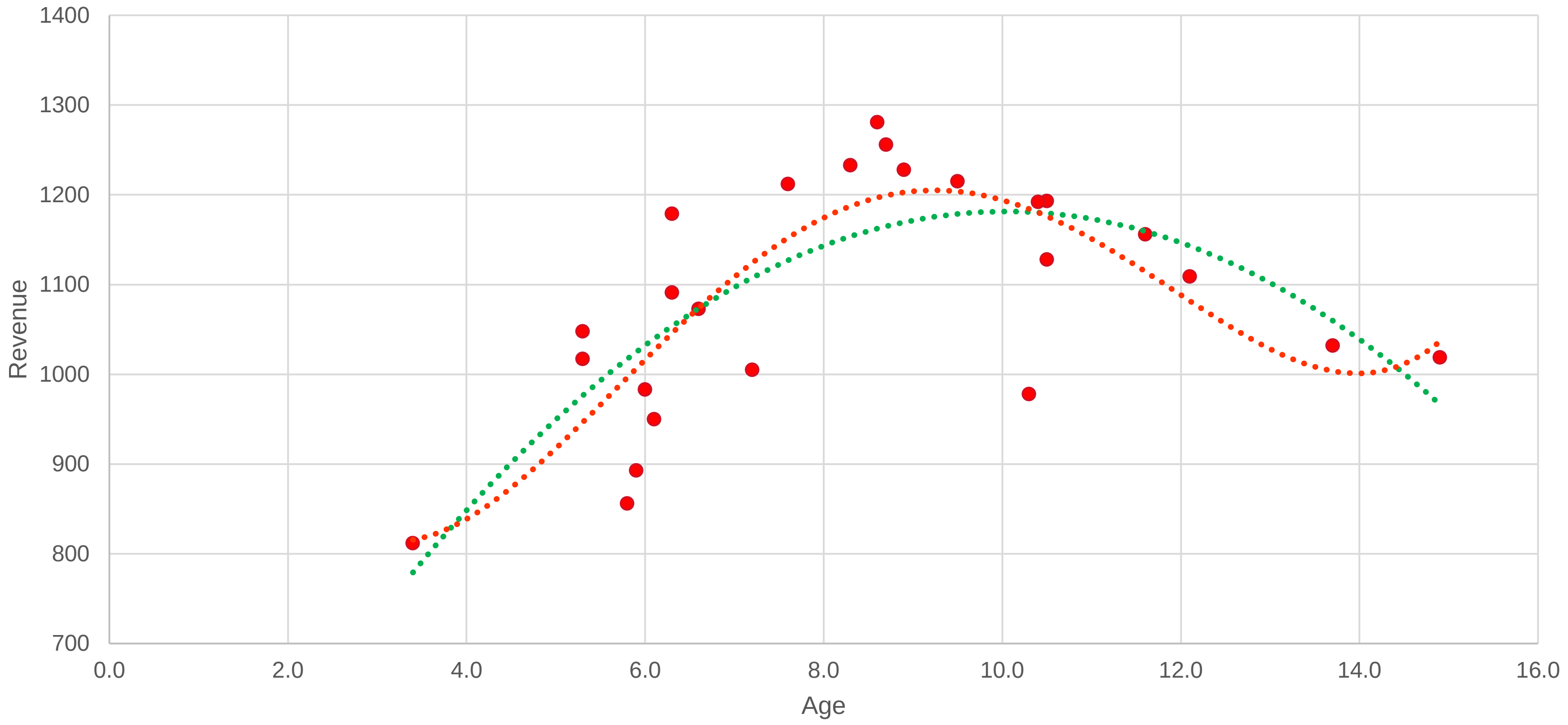
- Many respectable numerical packages, e.g., `sklearn.linear_model`, contain GLM implementation which includes logistic regression.

Regularization Intuition (QBUS6810)

THE UNIVERSITY OF SYDNEY

What Have We Learnt?

Supervised learning with continuous response- regression single or multiple features



$f(\mathbf{x}, \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$ $\Rightarrow$ **Just right**

$f(\mathbf{x}, \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4$ $\Rightarrow$ **Overfitting**

Regularization

$$f(\mathbf{x}, \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 \boxed{+\beta_3\, x_1^3 + \beta_4\, x_1^4}$$

Can we have a way to penalize parameters β_3 and β_4 to be close to 0, so that the model is approximately:

$$f(\mathbf{x}, \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$$

$$L(\boldsymbol{\beta}) = \frac{1}{2N} \sum_{n=1}^{N} (t_n - f(\mathbf{x}_n, \boldsymbol{\beta}))^2 \boxed{+ \lambda \beta_3^2 + \lambda \beta_4^2}$$

If λ were very large, e.g., 10,000, then parameters β_3 and β_4 would be heavily penalized, e.g., close to 0

Regularized Linear Regressions (QBUS6810)

Ridge Regression

The **Ridge Regression Estimator** is the minimiser of the cost function with quadratic regularization term

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n, \boldsymbol{\beta}))^2 + \frac{\lambda}{2N}\sum_{j=1}^{d}\beta_j^2$$

$\lambda \geq 0$ is a regularization parameter which regulates the tradeoff (regulates model complexity).

The penalised term does not include the intercept term β_0

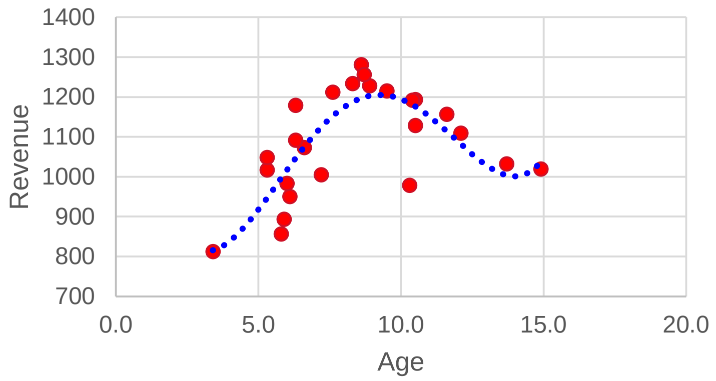
$\lambda = 0$, we have the ordinary linear regression cost function.

$\lambda = 0$ corresponds to the greatest complexity (bias is a minimum, but variance is high)

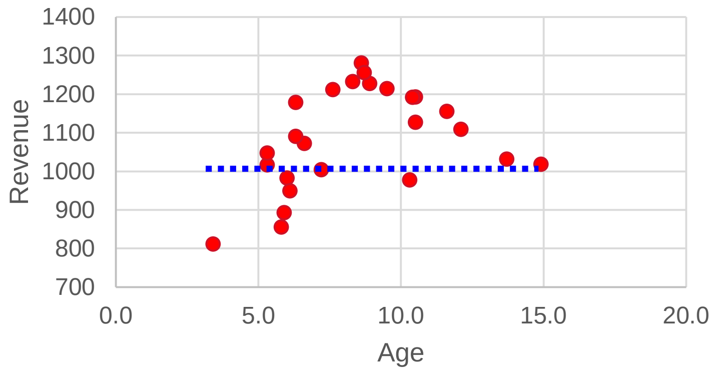
The penalty term penalises the departure from zero of the regression parameter i.e. shrinks them toward zero.

Ridge regression cannot zero out a specific coefficient. The model either ends up including all the coefficients in the model, or none of them

Small λ : no or low regularization. Can fit a high order polynomial model or complex model.



Large λ : high regularization. $\boldsymbol{\beta}$ will be small. If λ is very, very large, model becomes a horizontal line to the data.



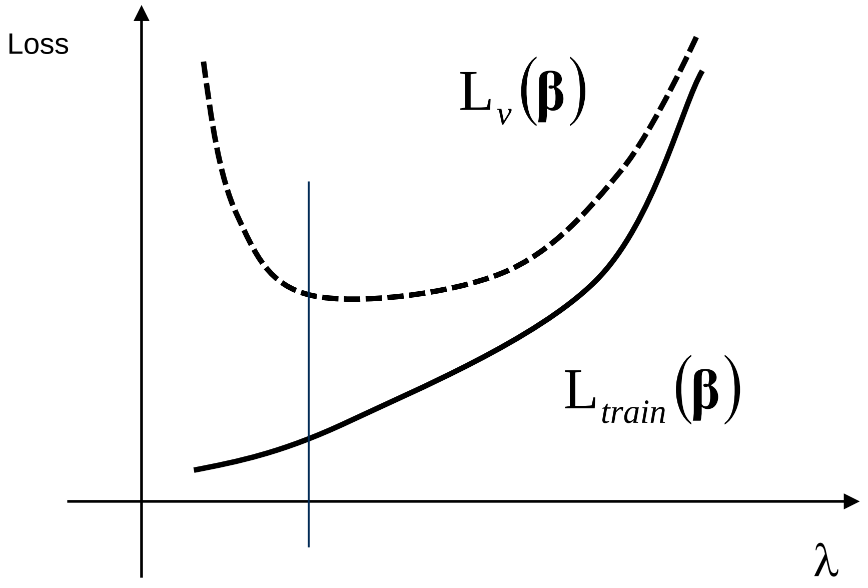
Learning Curve

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n,\boldsymbol{\beta}))^2 + \frac{\lambda}{2N}\sum_{j=1}^{d}\beta_j^2$$

This loss function is used estimate the model

Tend to **Overfitting**.
High variance.

Tend to **underfitting**.
High bias.



Best Model

$$L_{\text{train}}(\boldsymbol{\beta}) = \frac{1}{2N_{tr}}\sum_{n=1}^{N_{tr}}(t_n - f(\mathbf{x}_n,\boldsymbol{\beta}))^2$$

$$L_{\text{v}}(\boldsymbol{\beta}) = \frac{1}{2N_v}\sum_{n=1}^{N_v}(t_n - f(\mathbf{x}_n,\boldsymbol{\beta}))^2$$

How to Choose λ?

λ		Validation Loss
0.0001	Estimated	
0.001	Estimated	
0.01	Estimated	
0.02	Estimated	
0.04	Estimated	Smallest
…		
100	Estimated	

Use this model for test set

$$\min_{\boldsymbol{\beta}} \quad L(\boldsymbol{\beta})$$

Test a large number of different λ value, e.g., 10000 values between 0.0001 and 100, denoted by λ_j $(j = 1,2,3, \dots, 10000)$

Ridge Regression Gradient Descent

- Have some random starting points for all β_i;
- Keep updating all β_i (simultaneously) to decrease the loss function $L(\boldsymbol{\beta})$ value;
- Repeat until achieving minimum (convergence).

Partial derivative calculation omitted

$$\beta_0 := \beta_0 - \alpha \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_0} = \beta_0 - \alpha \frac{1}{N} \sum_{n=1}^{N} (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d - t_n)$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_1} = \beta_1 - \alpha \left[\frac{1}{N} \sum_{n=1}^{N} (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d - t_n) x_{n1} + \frac{\lambda}{N} \beta_1 \right]$$

$$\boxed{\boldsymbol{\beta} := \boldsymbol{\beta} - \alpha \frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \alpha \left[\frac{1}{N} \mathbf{X}^T (f(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{t}) + \frac{\lambda}{N} \boldsymbol{\beta} \right]}$$

Update simultaneously

$$\beta_d := \beta_d - \alpha \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_d} = \beta_d - \alpha \left[\frac{1}{N} \sum_{n=1}^{N} (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d - t_n) x_{nd} + \frac{\lambda}{N} \beta_d \right]$$

THE UNIVERSITY OF SYDNEY

The LASSO

Least absolute shrinkage & selection operator

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n, \boldsymbol{\beta}))^2 + \frac{\lambda}{2N}\sum_{j=1}^{d}|\beta_j|$$

- ❑ LASSO does both parameter shrinkage and variable selection automatically.
- ❑ Some coefficients are forced to zero as λ increases (effectively a subset selection)

Elastic Net

Elastic net is a regularized regression method that linearly combines the penalties of the lasso and ridge methods.

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n, \boldsymbol{\beta}))^2 + \frac{\lambda_1}{2N}\sum_{j=1}^{d}|\beta_j| + \frac{\lambda_2}{2N}\sum_{j=1}^{d}\beta_j^2$$

Note that due to the shared L1/L2 regularisation of Elastic-Net it does not aggressively prune features like Lasso. In practice it often performs well when used for regression prediction.

See Lecture03_Example01.py

CV with Regularization

- Suppose we have 150 data points and wish to find a better ridge regression. That is to find an appropriate λ for the ridge regression model.
- Under K-CV, we are going to test a large number of different λ values, e.g., 10000 values between 0.0001 and 100, denoted by λ_j $(j = 1,2,3, \ldots, 10000)$
- Divide (randomly) 150 data points into 5 groups, each with 30 data points

CV with Regularization

- We will then run the 5-fold cross validation with following steps for each λ_j
- For each λ_j, output mean validation error $(\mathrm{L_v}(\boldsymbol{\beta_1}) + \mathrm{L_v}(\boldsymbol{\beta_2})+\ldots+\mathrm{L_v}(\boldsymbol{\beta_5}))/5$ on validation sets and select the model with λ that generates the least error, say λ_{151}
- Then build the model with $\lambda_{151}$by

$$L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{n=1}^{N}(t_n - f(\mathbf{x}_n, \boldsymbol{\beta}))^2 + \frac{\lambda_{151}}{2N}\sum_{j=1}^{d}\beta_j^2$$

This process can be incorporated with LASSO and Elastic net as well.

Appropriate K in CV?

The special case K = N is known as the **leave-one-out (LOO) cross-validation**

With K = N, the cross-validation estimator is approximately unbiased for the true (expected) prediction error, but can have high variance because the K=N "training sets" are so similar to one another.

The computational burden is also considerable, requiring m applications of the learning method for LOO

On the other hand, with K = 5 say, cross-validation has lower variance, while bias could be a problem, depending on how the performance of the learning method varies with the size of the training set.

Overall, **five-fold or ten-fold** cross-validation are recommended as a good compromise

Feature Extraction and Representation

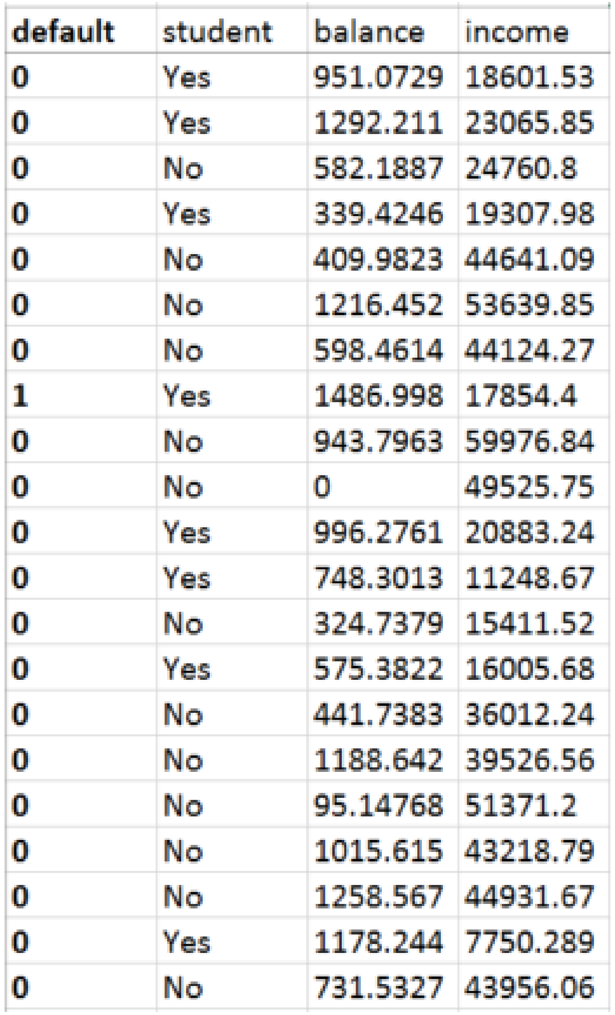
THE UNIVERSITY OF SYDNEY

Processing Features

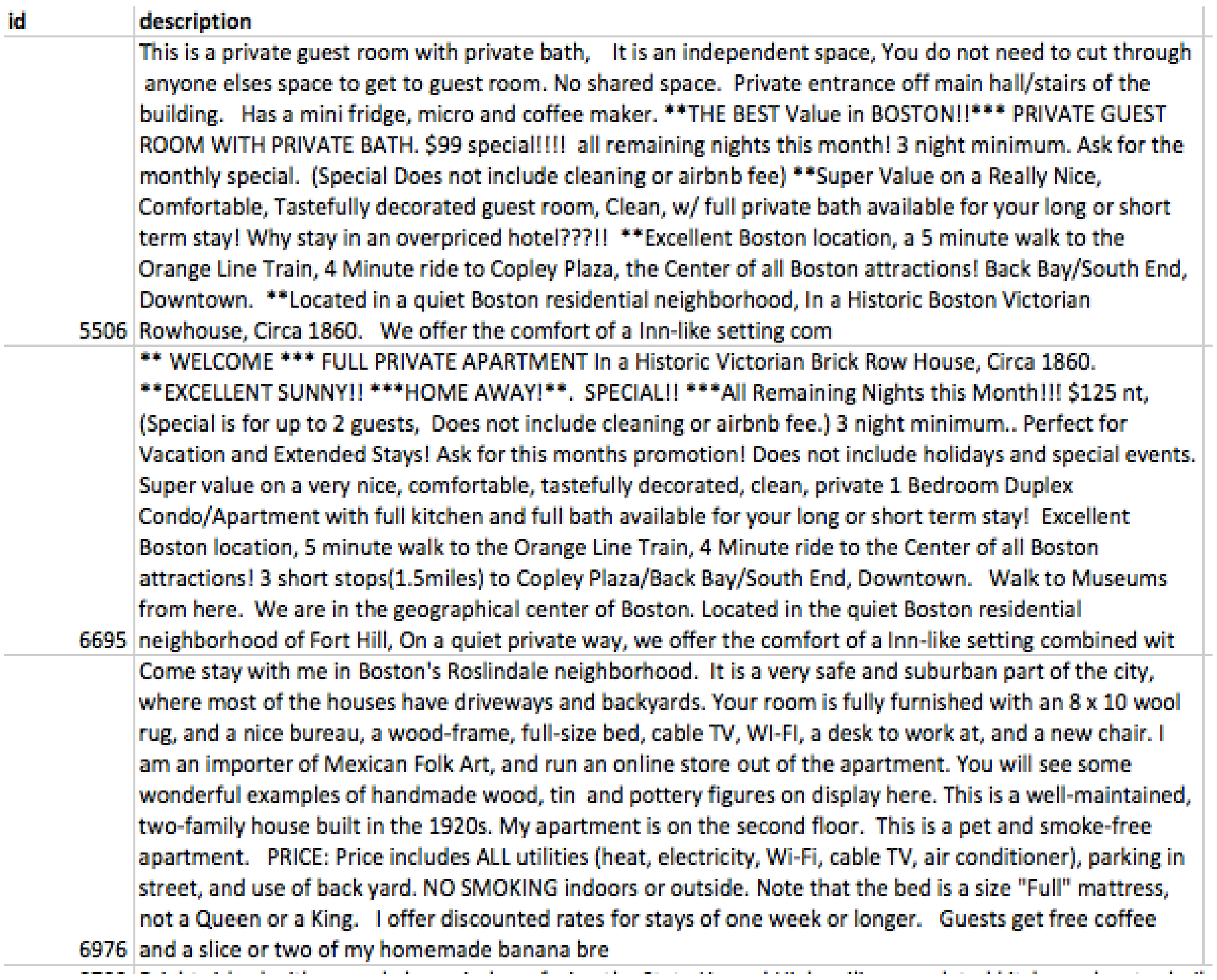
- All we have assumed so far is that data come to us in good shape and most of them are in numeric format and possibly in a categorical form
- In Python machine learning, we normally organise data into a matrix (or multidimensional arrays)
- However data coming from application domains could be in any forms or categories
- For business applications, we may have data in the form of text (or natural language), or in media such as audio and videos
- It is easy to deal with numeric data which can be sent to a machine learning straightaway

THE UNIVERSITY OF SYDNEY

How the real data look like?



default	student	balance	income
0	Yes	951.0729	18601.53
0	Yes	1292.211	23065.85
0	No	582.1887	24760.8
0	Yes	339.4246	19307.98
0	No	409.9823	44641.09
0	No	1216.452	53639.85
0	No	598.4614	44124.27
1	Yes	1486.998	17854.4
0	No	943.7963	59976.84
0	No	0	49525.75
0	Yes	996.2761	20883.24
0	Yes	748.3013	11248.67
0	No	324.7379	15411.52
0	Yes	575.3822	16005.68
0	No	441.7383	36012.24
0	No	1188.642	39526.56
0	No	95.14768	51371.2
0	No	1015.615	43218.79
0	No	1258.567	44931.67
0	Yes	1178.244	7750.289
0	No	731.5327	43956.06

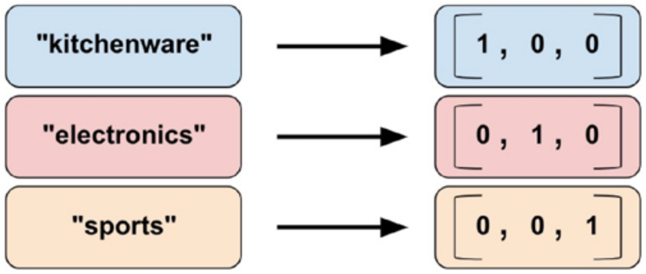


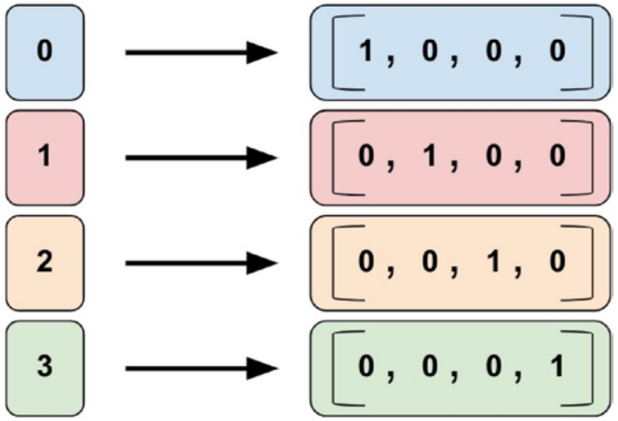
id	description
5506	This is a private guest room with private bath, It is an independent space, You do not need to cut through anyone elses space to get to guest room. No shared space. Private entrance off main hall/stairs of the building. Has a mini fridge, micro and coffee maker. **THE BEST Value in BOSTON!!*** PRIVATE GUEST ROOM WITH PRIVATE BATH. $99 special!!!! all remaining nights this month! 3 night minimum. Ask for the monthly special. (Special Does not include cleaning or airbnb fee) **Super Value on a Really Nice, Comfortable, Tastefully decorated guest room, Clean, w/ full private bath available for your long or short term stay! Why stay in an overpriced hotel???!! **Excellent Boston location, a 5 minute walk to the Orange Line Train, 4 Minute ride to Copley Plaza, the Center of all Boston attractions! Back Bay/South End, Downtown. **Located in a quiet Boston residential neighborhood, In a Historic Boston Victorian Rowhouse, Circa 1860. We offer the comfort of a Inn-like setting com
6695	** WELCOME *** FULL PRIVATE APARTMENT In a Historic Victorian Brick Row House, Circa 1860. **EXCELLENT SUNNY!! ***HOME AWAY!**. SPECIAL!! ***All Remaining Nights this Month!!! $125 nt, (Special is for up to 2 guests, Does not include cleaning or airbnb fee.) 3 night minimum.. Perfect for Vacation and Extended Stays! Ask for this months promotion! Does not include holidays and special events. Super value on a very nice, comfortable, tastefully decorated, clean, private 1 Bedroom Duplex Condo/Apartment with full kitchen and full bath available for your long or short term stay! Excellent Boston location, 5 minute walk to the Orange Line Train, 4 Minute ride to the Center of all Boston attractions! 3 short stops(1.5miles) to Copley Plaza/Back Bay/South End, Downtown. Walk to Museums from here. We are in the geographical center of Boston. Located in the quiet Boston residential neighborhood of Fort Hill, On a quiet private way, we offer the comfort of a Inn-like setting combined wit
6976	Come stay with me in Boston's Roslindale neighborhood. It is a very safe and suburban part of the city, where most of the houses have driveways and backyards. Your room is fully furnished with an 8 x 10 wool rug, and a nice bureau, a wood-frame, full-size bed, cable TV, WI-FI, a desk to work at, and a new chair. I am an importer of Mexican Folk Art, and run an online store out of the apartment. You will see some wonderful examples of handmade wood, tin and pottery figures on display here. This is a well-maintained, two-family house built in the 1920s. My apartment is on the second floor. This is a pet and smoke-free apartment. PRICE: Price includes ALL utilities (heat, electricity, Wi-Fi, cable TV, air conditioner), parking in street, and use of back yard. NO SMOKING indoors or outside. Note that the bed is a size "Full" mattress, not a Queen or a King. I offer discounted rates for stays of one week or longer. Guests get free coffee and a slice or two of my homemade banana bre

THE UNIVERSITY OF SYDNEY

Categorical Features

- In raw data, they are represented by strings. For example “Red”, “Green”, “Blue”, “Yellow”, and “White”.
- They are not suitable to machine learning. We need engineer them into numeric numbers.
  - Label Representation: Label representation: For example “Red” to 4, “Green” to 3, “Blue” to 2, “Yellow” to 1, and “White” to 0.
  - One-hot Encoding:





Use scikit-learn to make this transform or pandas’ `get_dummies`:

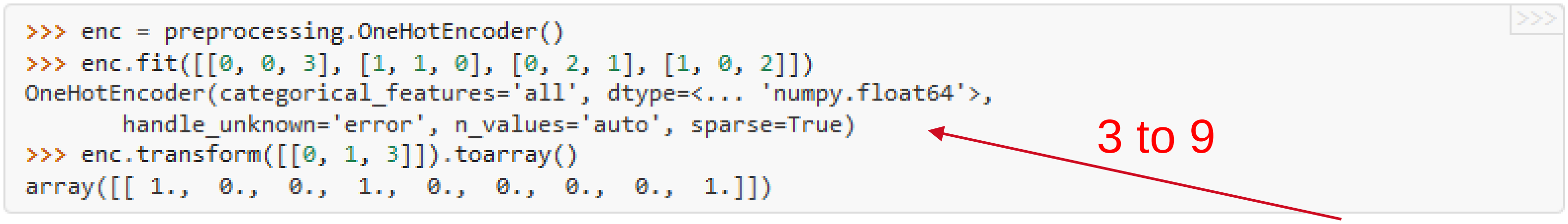
`Lecture03_Example02.py`

THE UNIVERSITY OF SYDNEY

Transforming Categorical Features in scikit-learn

❑ Encoding categorical features

❖ Converting a categorical feature to one-hot coding by OneHotEncoder

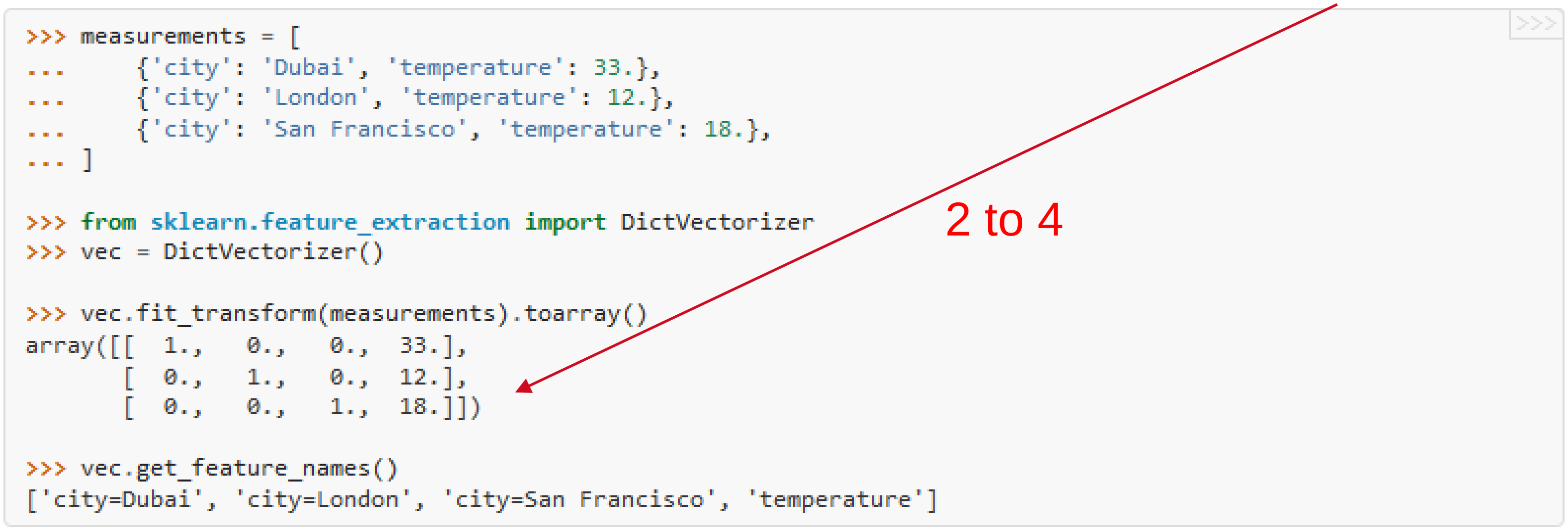


```
>>> enc = preprocessing.OneHotEncoder()
>>> enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])
OneHotEncoder(categorical_features='all', dtype=<... 'numpy.float64'>,
       handle_unknown='error', n_values='auto', sparse=True)
>>> enc.transform([[0, 1, 3]]).toarray()
array([[ 1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.]])
```

3 to 9

The number of features increased

❑ Loading features from dicts



```
>>> measurements = [
...     {'city': 'Dubai', 'temperature': 33.},
...     {'city': 'London', 'temperature': 12.},
...     {'city': 'San Francisco', 'temperature': 18.},
... ]

>>> from sklearn.feature_extraction import DictVectorizer
>>> vec = DictVectorizer()

>>> vec.fit_transform(measurements).toarray()
array([[  1.,   0.,   0.,  33.],
       [  0.,   1.,   0.,  12.],
       [  0.,   0.,   1.,  18.]])

>>> vec.get_feature_names()
['city=Dubai', 'city=London', 'city=San Francisco', 'temperature']
```

2 to 4

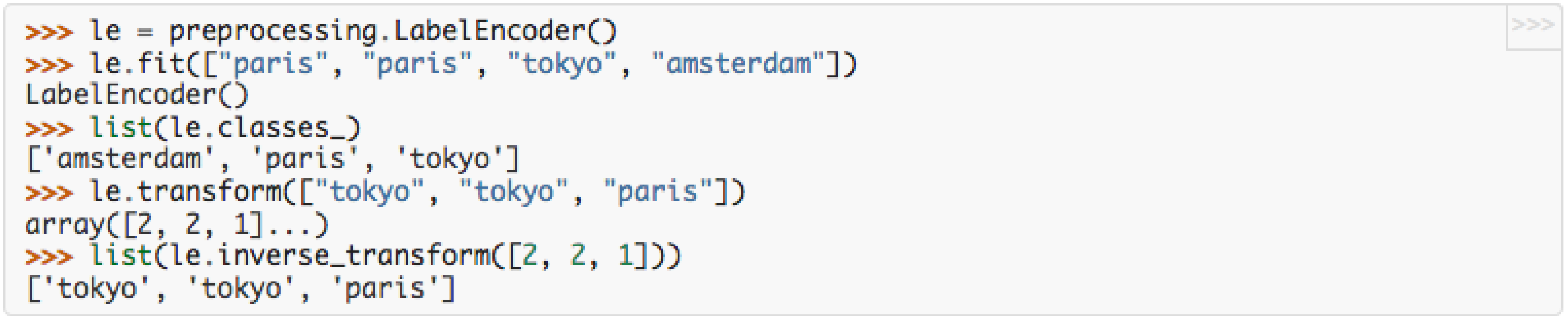
from scikit-learn documentation

Ordinal Features

- In raw data, they are represented by strings. For example “Strongly Agree”, “Agree”, “Neutral”, “Disagree”, and “Strongly Disagree”.

- The order information is important for modelling. Most time, we can encode such feature in terms of integers, such as “Strongly Agree” to 5, “Agree” to 4, “Neutral” to 3, “Disagree” to 2, and “Strongly Disagree” to 1.

- LabelEncoder in scikit-learn can be used to transform such non-numerical labels to numerical labels

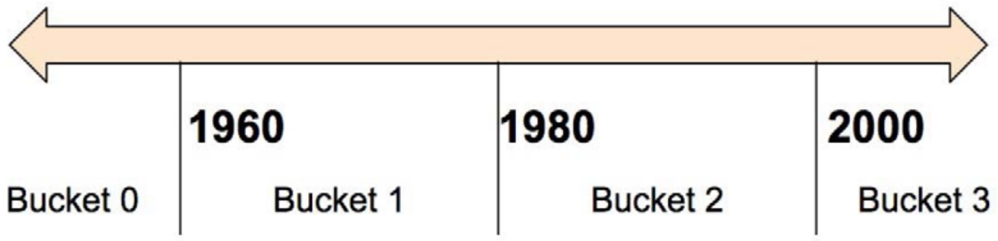


```
>>> le = preprocessing.LabelEncoder()
>>> le.fit(["paris", "paris", "tokyo", "amsterdam"])
LabelEncoder()
>>> list(le.classes_)
['amsterdam', 'paris', 'tokyo']
>>> le.transform(["tokyo", "tokyo", "paris"])
array([2, 2, 1]...)
>>> list(le.inverse_transform([2, 2, 1]))
['tokyo', 'tokyo', 'paris']
```

from scikit-learn documentation

Bucketized Feature (in Tensorflow)

- Some times it is more meaningful to convert numbers into numerical ranges
- Thus we shall engineer some numeric features into categorical feature



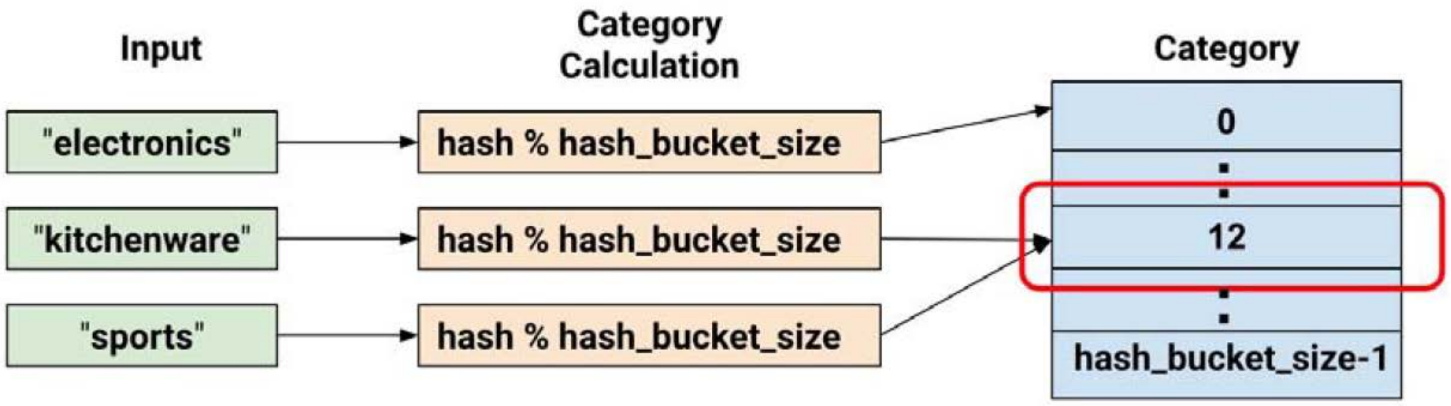
Dividing year data into four buckets.

- Then convert to one-hot coding or labels

Date Range	Represented as...
< 1960	[1, 0, 0, 0]
>= 1960 but < 1980	[0, 1, 0, 0]
>= 1980 but < 2000	[0, 0, 1, 0]
> 2000	[0, 0, 0, 1]

Feature Hashing

- If a categorical feature has huge of different values, then the one-hot coding is a long (sparse) vector
- Instead of using a long 0-1 vector, we use a hash function which calculates a hash code. We are forcing the different input values to a smaller set of categories



- Collision: Both “kitchenware” and “sports” may be mapped to the same values

```
hasher = FeatureHasher(input_type='string')
X = hasher.transform(raw_X)
```

Bag-of-Words

- In Business Intelligence, we analyse texts such business plan, business report, even news.
- Machine learning algorithms cannot work with raw text directly and the text must be converted in to numbers.
- The BoW representation of text describes the occurrence of words within a document
- For example, suppose we have a vocabulary of 1000 words. We have a document in which ``ours" appears 3 times, ``competition'' appears 1 time and ``managers'' 10 times.
- We will represent this document as a vector of dimension 1000 (such as each component in the vector corresponds to a word in the vocabulary) such that 3 will be in the position of the vector corresponding to ``ours'', 1 at the position corresponding to `` competition'' and 15 at the position corresponding ``managers''.
- All other positions have values of 0. So the vector looks like

$$x = (0, \dots, 0, 3, 0, \dots, 0, 1, 0, \dots, 15, 0, \dots, 0)^T \in \mathbb{R}^{1000}$$

which is sparse. Why?

Bag-of-Words

- Sciki-learn can extract numerical features from text content such as counting the occurrence of tokens in each document

- A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus

- See Lecture03_Example03.py

Text Feature Extraction

There are other definitions for these two "frequencies"

- **tf-idf Term Weighting**
  - tf-idf(t, d) is defined as

$$\mathrm{tf{-}idf}(t,d) = \mathrm{tf}(t,d) \times \mathrm{idf}(t)$$

  - tf(*t*,*d*) (term frequency): is the frequency of a term *t* in a document *d*, i.e., the occurring count number of *t* in *d*

  - idf(*t*) **inverse document frequency:** a measure of how much information the word *t* provides, that is, whether the term is common or rare across all documents

$$\mathrm{idf}(t) = \log\left(\frac{\text{the total of number of documents}}{\text{the number of documents in corpus containing term } t}\right)$$

- How is this done in scikit-learn? Lecture03_Example04.py

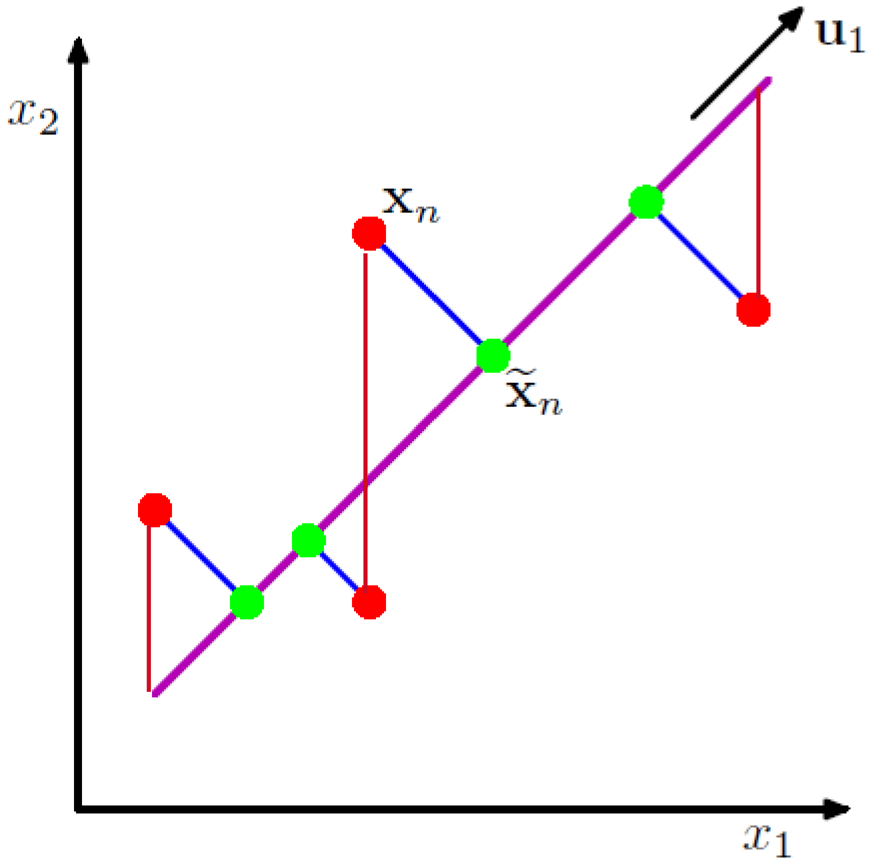
Embedding Representation

- Both One-hot Encoding and BoW may produce feature representations which are of large dimension and sparse.
- High dimension will result in the so-called curse of dimensionality problem in many machine learning algorithms.
- As those representations are actually sparse, so a natural question is whether we can find a compact format for these sparse representations.
- The so-called learning embedding or dimensionality reduction can achieve this goal

Principle Component Analysis (PCA)

- Principal component analysis seeks a space of lower dimensionality, known as the principal subspace and denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots).

- How can we find this line? Can we do this by linear regression?



Principle Component Analysis (PCA)

- Objective: given a set of *d* measurements on *N* individuals, we aim at determining $r \leq d$ orthogonal (uncorrelated) variables, called principal components, defined as linear combinations of the original ones

- The PCs are uncorrelated and have decreasing variance
  - Synthesis: information dimensionality reduction
  - Interpretation: express the original data in terms of a reduced number of underlying variables (factors)
  - Score the individual proles, with a summary score
  - Obtain multivariate displays (scatterplot) of the units in two or three dimensions

- The first component is designed to capture as much of the variability in the data as possible, and the succeeding components in turn extract as much of residual variability as possible

PCA: The Algorithm

Size $N \times d$

- Given a set of data $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$. Suppose they have been centralised, i.e., removing the mean from them. Collect them in a data matrix $\mathbf{X}$

Size $d \times d$

- Calculate the variance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$
- Conduct the eigen-decomposition of $\mathbf{S}$ such that

$$\mathbf{S} = \mathbf{A}\Lambda\mathbf{A}^T$$

where $\mathbf{A}^\mathbf{T}\mathbf{A} = \mathbf{I}_\mathbf{d}$ and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d)$.

- The first r $(r \leq d)$ principle components of $\mathbf{X}$ are given by

Size $N \times r$

$$\mathbf{Z}_r = \mathbf{X}\mathbf{A}_r$$

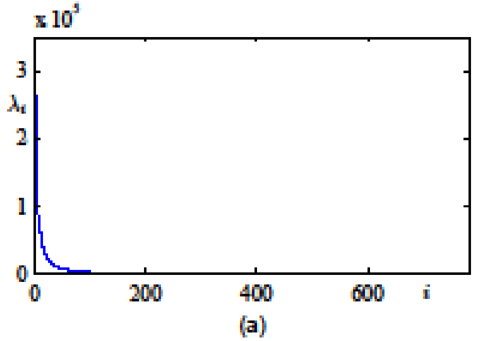
where $\mathbf{A}_r$ is the matrix of the first r columns of $\mathbf{A}$.

- Each row of $\mathbf{Z}_r$ (in r new factors/features) is a new representation of the given data, i.e., the corresponding row in $\mathbf{X}$ (in d attributes/features)

PCA: Selecting r

- Consider the share of the total variance absorbed by the first r components $\mathbf{Z}_r$

$$Q_r = \frac{\sum_{h=1}^{r} \lambda_h}{\sum_{h=1}^{d} \lambda_h}$$



Select r so that $Q_r \geq 0.95$ for example

- Kaiser criterion: computer the average eigenvalues

$$\bar{\lambda} = \frac{1}{d} \sum_{h=1}^{d} \lambda_h$$

- Select the first r components for which $\lambda_h > \bar{\lambda}$. Note: if the variables are standardised $\bar{\lambda} = 1$.

Python Example

(Lecture03_Example05.py)